

السلام عليكم و رحمة الله و بركاتُهُ.

هذا الدرس يتمركز حول التايمر ¤Timers¤ ماهُوَ ؟ مالفَائِدةُ منـهُ ؟ كَيفَ ؟ و لِمَـاذَا ؟

حيـث سندرس التايمر مــن الأسـاس و لـيس وضـع الأداة على الفورم و .. مبروك!!!

منتدى لغات البرمجة - بيت الإبداع

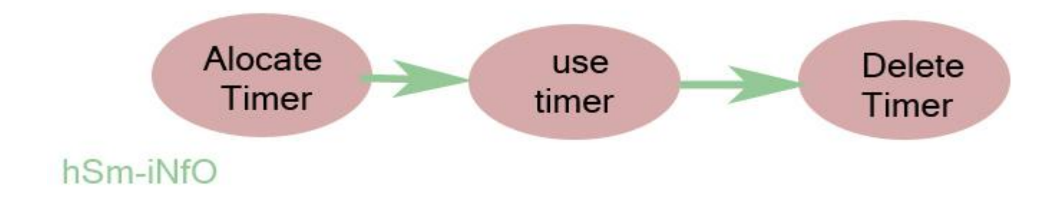
التـايمر ¤ المؤقت ¤ هـو وحدة توقيت يحتاج إليها المبرمج ,تستعمـل في الكثير من البرامج

و هي تخصيص مهلة زمنية يتم مع مَضِيِّهَا إرسال رسالة¤ القيـام بعمـل .¤ أي سرعة إرسال الرسالة.

التايمـر ¤ Timer ¤ يعمل بوحدة الميليثانية أي 1/1000 ثـانية و هي فترة صغيرة جـدا.

لكن كَكُلِّ للتـايمر حدود حيث أصغرهـا 1 ميليثانية و أكبرها 4.294.967.295 ميليثـانية و هو مايقارب 50 يوما!!!

التعامل مع التايمر يمر بــ3 مـراحل كما يوضحه المخطط:



منتدى لغات البرمجة - بيت الإبداع

المرحلة الأولـى: Alocate Timer

مرحـلة إنشاء التايمر ¤Timer¤ و حجز مكـان له في الذاكرة.

غالبا ما تكون هذه المرحلة عند إنشاء النافذة مثلا عنـد WM_INITDIALOG أو WM_CREATE..

فيتم الحجز مع ّإنشاء النـافذة , و هنـاك دالة مسؤولة عن ذلك هي SetTimer

الدالة تأخذ هذا الشكل:

```
UINT_PTR SetTimer(
    HWND hWnd,
    UINT_PTR nIDEvent,
    UINT uElapse,
    TIMERPROC lpTimerFunc
);
```

http://msdn.microsoft.com/en-us/library/ms644906%28VS.85%29.aspx

هذه الدالة من مكتبة User32.dll

و هي تتعامل مع وحدة حساب الوقت المتواجدة فـي *ROM BIOS* لكن بطريقة غير مباشرة حيث تحول

إلى دالة تعمل تحت Kernel Mode لكنه ليس موضوعنا اليوم , للدالةكما نلاحظ 4 بارامترات هي:

: hWnd وهو مقبض للنافذة التي نريد وضع التايمر بهـا

: nIDEvent وهو رقم التايمر الذي سنعرفه ¤ هذا البارامتر مهمجدا عند استخدام أكثر من 1 تايمر. ¤

: uElapse وتحدد قيمة المهلة , كمـا قلنا سـابقابالميليثانية.

: lpTimerFunc وهو مؤشر نحو الدالة.

المرحلة الثانية: use Timer

هذه المرحلة تخص إستعمال التايمر كما قلنا سابقا أي إرسال رسالة مع مرور المهلة

هذه الرسـالة تسمى WM_TIMER

و مع مرور فترة معينة يتم إعادة القيام بالوظائف عند تلك الرسالة

كمثـال نضع شرطا , و في الـInterval وضعنا 3000 ميليثانية , سيعيد البرنامج التحقق من الشرط مع مرور كل 3 ثوان.

المرحلة الثالثة: Delete Timer

هذه العملية ضرورية و تعني التخلص من التايمر.

التخلص من التايمر يكون إما بأمر من المبرمج أو بدونه.

حيث بأمر من المبرمج نعني بذلك أن المبرمج سيقوم بحذف التايمر عند الإنتهاء من وظيفته.

هنا سيستخدم دالة مسؤولة عن ذلك هي KillTimer

```
BOOL KillTimer(
    HWND hWnd,
    UINT_PTR uIDEvent
);
```

هذه الدالة كما نلاحظ تأخذ بارامترين هما:

: hWnd و هو مقبض للنافذة التي بها التايمر.

: uIDEvent و هو رقم التايمر الذي به تتعرف الدالة إليه , حسب ما حددناه سابقا.

http://msdn.microsoft.com/en-us/library/ms644903%28VS.85%29.aspx

قلنا سابقا أن التايمر يحذف في كلتا الحالتين سواء بأمر من المبرمج أو بدونه

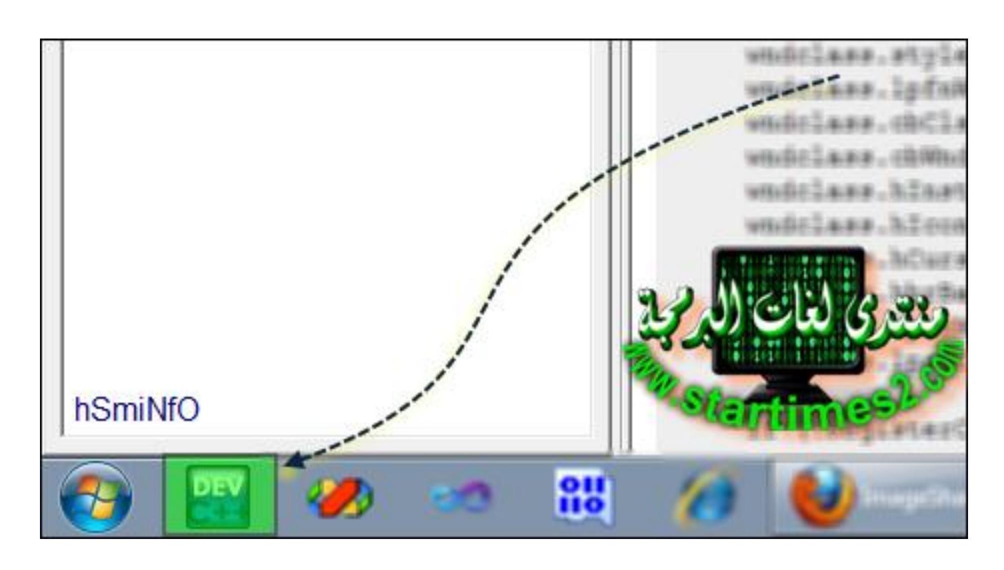
بدون المبرمج التايمر سيحذف تلقائيا عند مغادرة البرنامج و بالطبع سيستمل تلقائيا الدالة KillTimer

منتدى لغات البرمجة - بيت الإبداع

الآن بعد توضيح آلية عمل التايمر ¤ Timer ¤ هنـاك الكثِيرُ من الأشْيَاء التِّي يَلزَمُ معرفَتُهـا

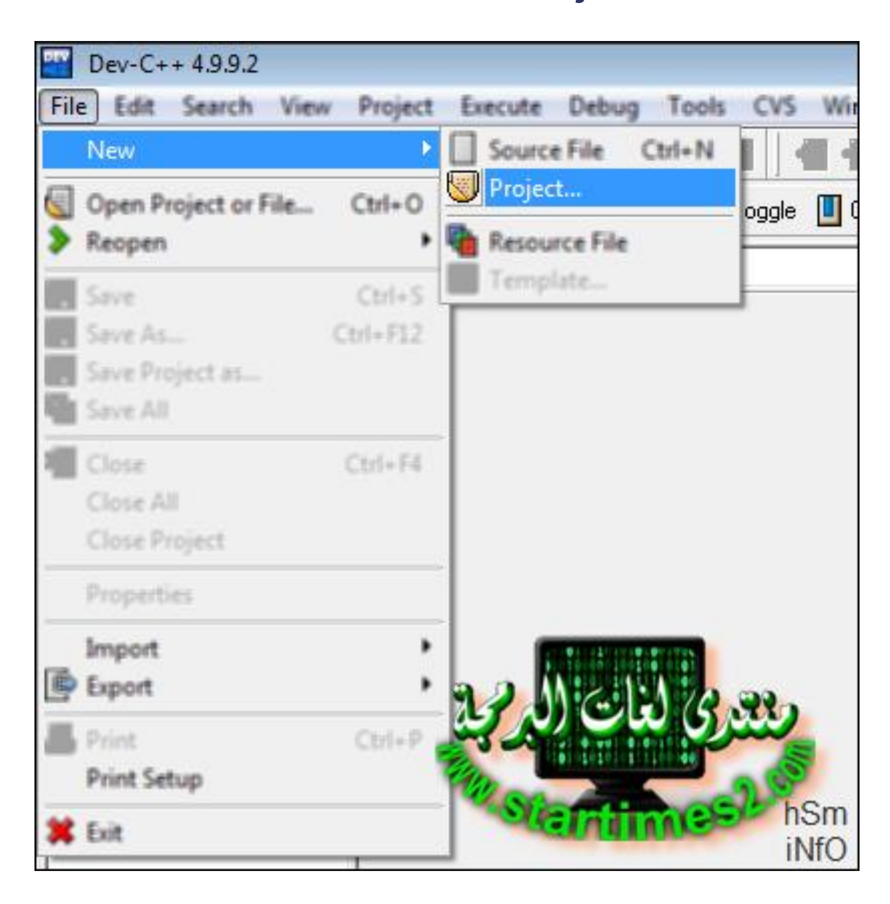
سنأتي أولا بمثال بسيط لكيفية استخدام التايمر مـع C++ طبعا😁

سنعمل على DevC++ ide لتسهيل الكثير من الأمور

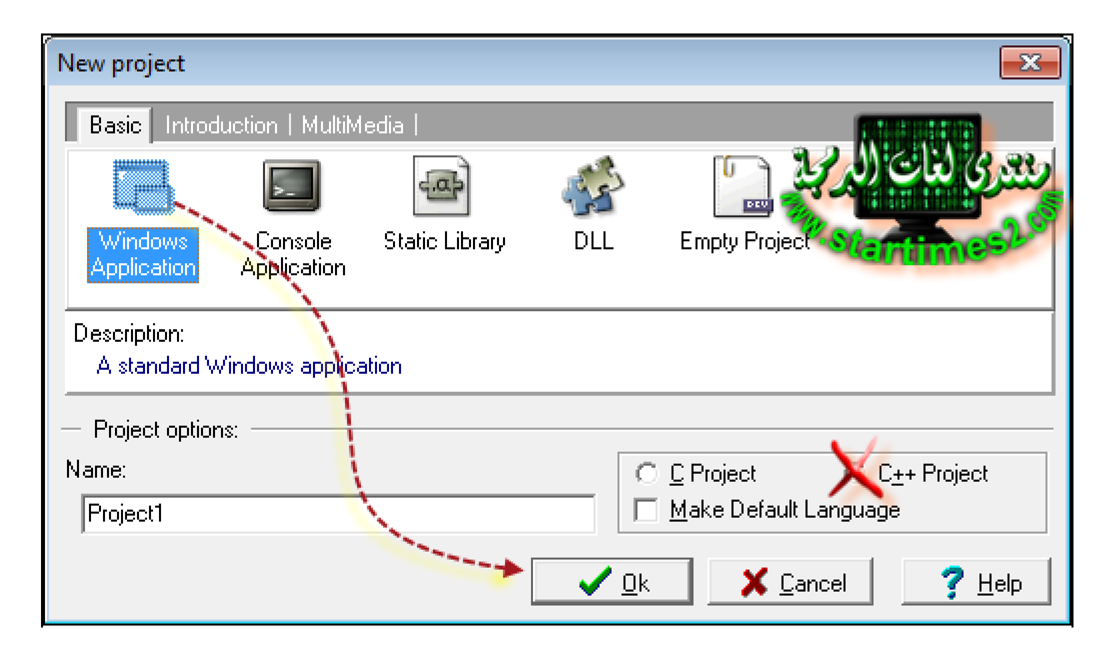


بعدها إذهب إلى

File==>New==>Project



بعدها قم بما يلي:



هذا سورس بسيط كـان مخزن مسبقا مع البرنامج

```
#include <windows.h>

/*  Declare Windows procedure  */
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);

/*  Make the class name into a global variable  */
char szClassName[ ] = "WindowsApp";

int WINAPI WinMain (HINSTANCE hThisInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpszArgument,
                    int nFunsterStil)

{
    HWND hwnd;               /* This is the handle for our window */
    MSG messages;            /* Here messages to the application are saved */
    WNDCLASSEX wincl;        /* Data structure for the windowclass */

    /* The Window structure */
    wincl.hInstance = hThisInstance;
    wincl.lpszClassName = szClassName;
    wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
    wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
    wincl.cbSize = sizeof (WNDCLASSEX);

    /* Use default icon and mouse-pointer */
    wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
    wincl.lpszMenuName = NULL;                 /* No menu */
    wincl.cbClsExtra = 0;                      /* No extra bytes after the window class */
    wincl.cbWndExtra = 0;                      /* structure or the window instance */
    /* Use Windows's default color as the background of the window */
    wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

    /* Register the window class, and if it fails quit the program */
    if (!RegisterClassEx (&wincl))
        return 0;

    /* The class is registered, let's create the program*/
    hwnd = CreateWindowEx (
           0,                   /* Extended possibilites for variation */
           szClassName,         /* Classname */
           "Windows App",       /* Title Text */
```

mpile Log | Debug | Find Results

لاحظ هذا المكـان في الأسفل

```
LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)                  /* handle the messages */
    {
        case WM_DESTROY:
            PostQuitMessage (0);       /* send a WM_QUIT to the message queue */
            break;
        default:                      /* for messages that we don't deal with */
            return DefWindowProc (hwnd, message, wParam, lParam);
    }

    return 0;
}
```

هذا ما يسمى بحلقة الرسائل حيث هنا قلب البرنـامج

نلاحظ وجود رسالة واحدة هي **WM_DESTROY** و هي خاصة بغلق البرنـامج

كما قلنـا سابقا سنضيف رسالتين

WM_CREATE و بداخلها سنضع دالة إنشاء التايمر**SetTimer**

WM_TIMER و بداخلها سنضع الكود الخاص بالتايمر

```
switch (message)                  /* handle the messages */
{
    case WM_CREATE:
            SetTimer(hwnd,0,3000,0);
            break;

    case WM_TIMER:
            MessageBox(hwnd,"This is a timer test !!","Information",MB_OK);
            break;

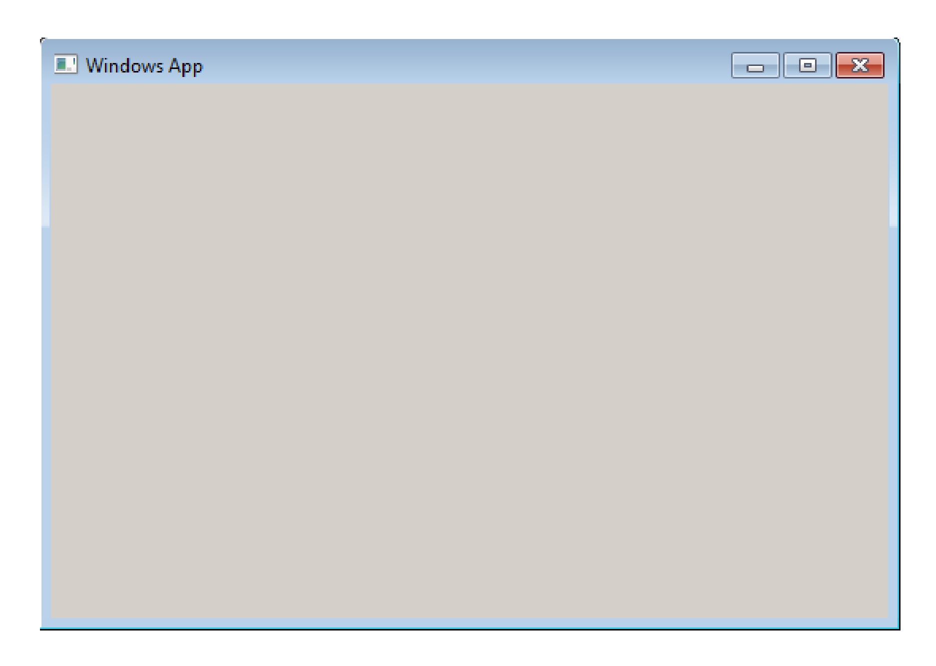
    case WM_DESTROY:
```

hSm_iNfO

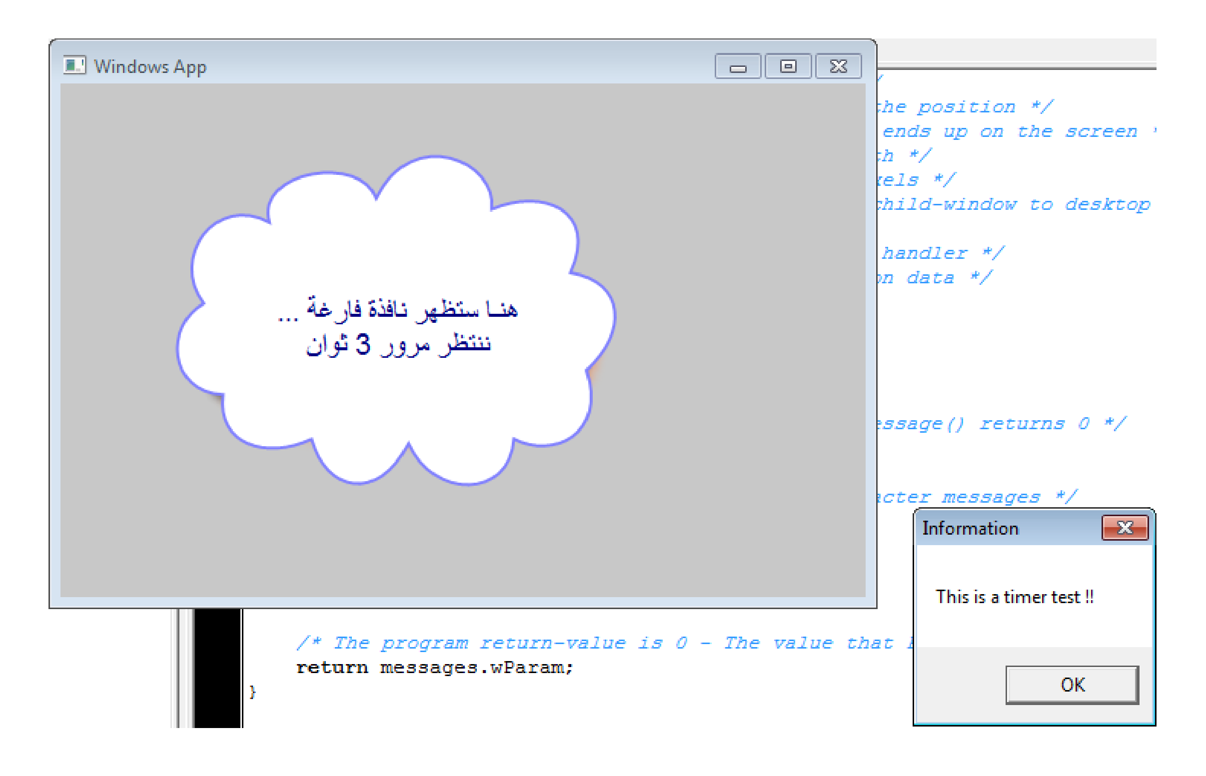
أظن كل شيء واضح إلى الآن

لاحظوا في المثال وضعت علبة تظهر مع مرور كل3 ثوان

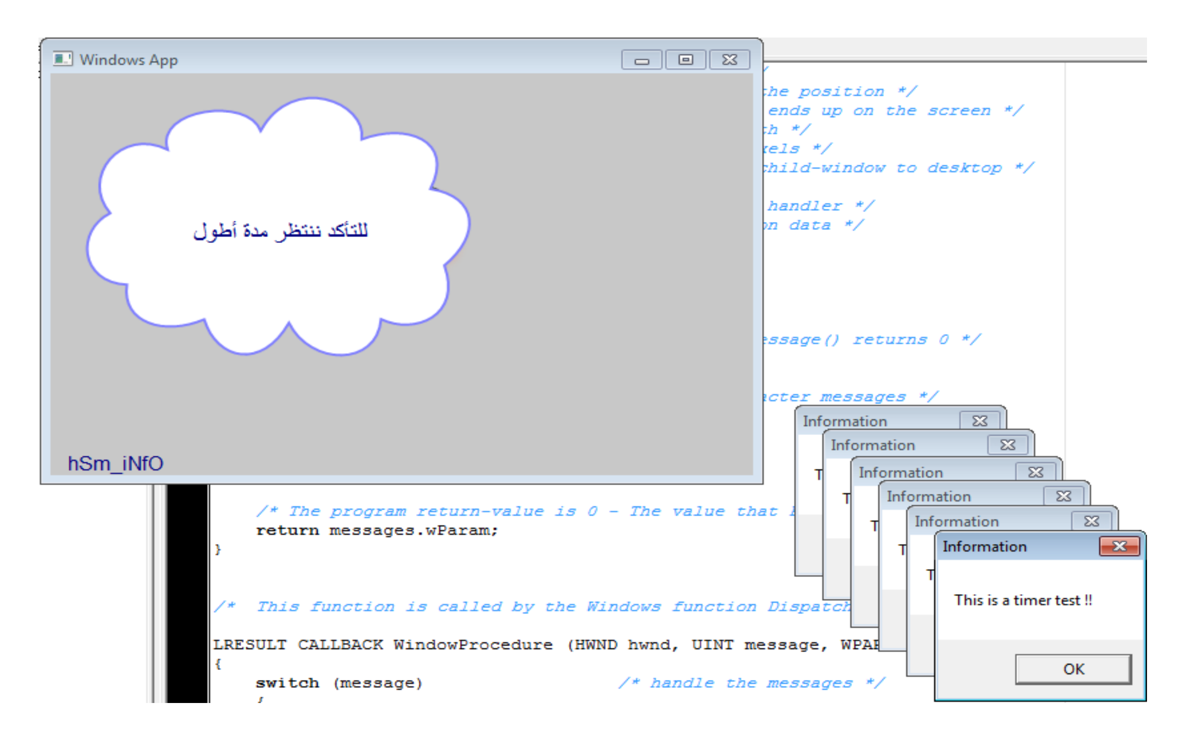
النتيجة ستكون نافذة عادية



بعد مرور 3 ثوان



فقط للتأكد ننتظر مدة أطول



هنا أكون قد وضعت مثالا بسيطا للتوضيح

ملاحظة : كما قلنا سابقا لم نتخلص من التايمر لذا فسيتخلص منه البرنامج عند الخروج

لكن يمكننا توقيف عمله بإضافة

KillTimer(hwnd,0);

منتدى لغات البرمجة - بيت الإبداع

السورس المستعمل في المثـال أعلاه

```
#include <windows.h>

LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);

char szClassName[ ] = "WindowsApp";

int WINAPI WinMain (HINSTANCE hThisInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpszArgument,
                    int nFunsterStil)

{
    HWND hwnd;               /* This is the handle for our window */
    MSG messages;            /* Here messages to the application are saved */
    WNDCLASSEX wincl;        /* Data structure for the windowclass */

    /* The Window structure */
    wincl.hInstance = hThisInstance;
    wincl.lpszClassName = szClassName;
    wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
    wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
    wincl.cbSize = sizeof (WNDCLASSEX);
```

```
    /* Use default icon and mouse-pointer */
    wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
    wincl.lpszMenuName = NULL;                 /* No menu */
    wincl.cbClsExtra = 0;                      /* No extra bytes after the window class */
    wincl.cbWndExtra = 0;                      /* structure or the window instance */
    /* Use Windows's default color as the background of the window */
    wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;

    /* Register the window class, and if it fails quit the program */
    if (!RegisterClassEx (&wincl))
        return 0;

    /* The class is registered, let's create the program*/
    hwnd = CreateWindowEx (
           0,                   /* Extended possibilites for variation */
           szClassName,         /* Classname */
           "Windows App",       /* Title Text */
           WS_OVERLAPPEDWINDOW, /* default window */
           CW_USEDEFAULT,       /* Windows decides the position */
           CW_USEDEFAULT,       /* where the window ends up on the screen */
           544,                 /* The programs width */
           375,                 /* and height in pixels */
           HWND_DESKTOP,        /* The window is a child-window to desktop */
           NULL,                /* No menu */
           hThisInstance,       /* Program Instance handler */
           NULL                 /* No Window Creation data */
           );

    /* Make the window visible on the screen */
    ShowWindow (hwnd, nFunsterStil);

    /* Run the message loop. It will run until GetMessage() returns 0 */
    while (GetMessage (&messages, NULL, 0, 0))
    {
        /* Translate virtual-key messages into character messages */
        TranslateMessage(&messages);
        /* Send message to WindowProcedure */
        DispatchMessage(&messages);
    }

      return messages.wParam;
}


/*  This function is called by the Windows function DispatchMessage()  */

LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)                  /* handle the messages */
    {

        case WM_CREATE:
             SetTimer(hwnd,0,3000,0);
             break;

        case WM_TIMER:
             MessageBox(hwnd,"This is a timer test !!","Information",MB_OK);
             break;

        case WM_DESTROY:
            PostQuitMessage (0);       /* send a WM_QUIT to the message queue */
            break;
        default:                      /* for messages that we don't deal with */
            return DefWindowProc (hwnd, message, wParam, lParam);
    }
```

```
    return 0;
}
```


الآن بقيـت طريقـة إستخدام تايمرين 2 أو أكثـر:

علينـا أولا تعريـف ثابتين يحملان الرقـم 1 و 2

```
#define TIMER_ONE 1

#define TIMER_TWO 2
```

بعدهـا علينـا تعريف التايمرين بإستخدام الدالة SetTimer

```
SetTimer (hwnd, TIMER_ONE, 3000, NULL) ;
SetTimer (hwnd, TIMER_TWO, 6000, NULL) ;
```

ثـم نتجـه إلى حلقة الرسائل و نغير كالتالي:

```
case WM_TIMER:
          switch (wParam)
          {
                case TIMER_ONE:
                    MessageBox(NULL,"This is Timer 1","Timer1",MB_OK);
                    break ;
                case TIMER_TWO:
                    MessageBox(NULL,"This is Timer 2","Timer2",MB_OK);
                    break ;
                     }
```

ليكون إجمالي ماكتبناه

```
#include <windows.h>
#define TIMER_ONE 1
#define TIMER_TWO 2
/*  Declare Windows procedure  */
LRESULT CALLBACK WindowProcedure (HWND, UINT, WPARAM, LPARAM);
/*  Make the class name into a global variable  */
char szClassName[ ] = "WindowsApp";
int WINAPI WinMain (HINSTANCE hThisInstance,
                    HINSTANCE hPrevInstance,
                    LPSTR lpszArgument,
                    int nFunsterStil)
{
    HWND hwnd;               /* This is the handle for our window */
    MSG messages;            /* Here messages to the application are saved */
    WNDCLASSEX wincl;        /* Data structure for the windowclass */
    /* The Window structure */
    wincl.hInstance = hThisInstance;
    wincl.lpszClassName = szClassName;
    wincl.lpfnWndProc = WindowProcedure;      /* This function is called by windows */
    wincl.style = CS_DBLCLKS;                 /* Catch double-clicks */
    wincl.cbSize = sizeof (WNDCLASSEX);
```

```
    /* Use default icon and mouse-pointer */
    wincl.hIcon = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hIconSm = LoadIcon (NULL, IDI_APPLICATION);
    wincl.hCursor = LoadCursor (NULL, IDC_ARROW);
    wincl.lpszMenuName = NULL;                 /* No menu */
    wincl.cbClsExtra = 0;                      /* No extra bytes after the window class */
    wincl.cbWndExtra = 0;                      /* structure or the window instance */
    /* Use Windows's default color as the background of the window */
    wincl.hbrBackground = (HBRUSH) COLOR_BACKGROUND;
    /* Register the window class, and if it fails quit the program */
    if (!RegisterClassEx (&wincl))
        return 0;
    /* The class is registered, let's create the program*/
    hwnd = CreateWindowEx (
           0,                   /* Extended possibilites for variation */
           szClassName,         /* Classname */
           "Windows App",       /* Title Text */
           WS_OVERLAPPEDWINDOW, /* default window */
           CW_USEDEFAULT,       /* Windows decides the position */
           CW_USEDEFAULT,       /* where the window ends up on the screen */
           544,                 /* The programs width */
           375,                 /* and height in pixels */
           HWND_DESKTOP,        /* The window is a child-window to desktop */
           NULL,                /* No menu */
           hThisInstance,       /* Program Instance handler */
           NULL                 /* No Window Creation data */
           );
    /* Make the window visible on the screen */
    ShowWindow (hwnd, nFunsterStil);
    /* Run the message loop. It will run until GetMessage() returns 0 */
    while (GetMessage (&messages, NULL, 0, 0))
    {
        /* Translate virtual-key messages into character messages */
        TranslateMessage(&messages);
        /* Send message to WindowProcedure */
        DispatchMessage(&messages);
    }
    /* The program return-value is 0 - The value that PostQuitMessage() gave */
    return messages.wParam;
}

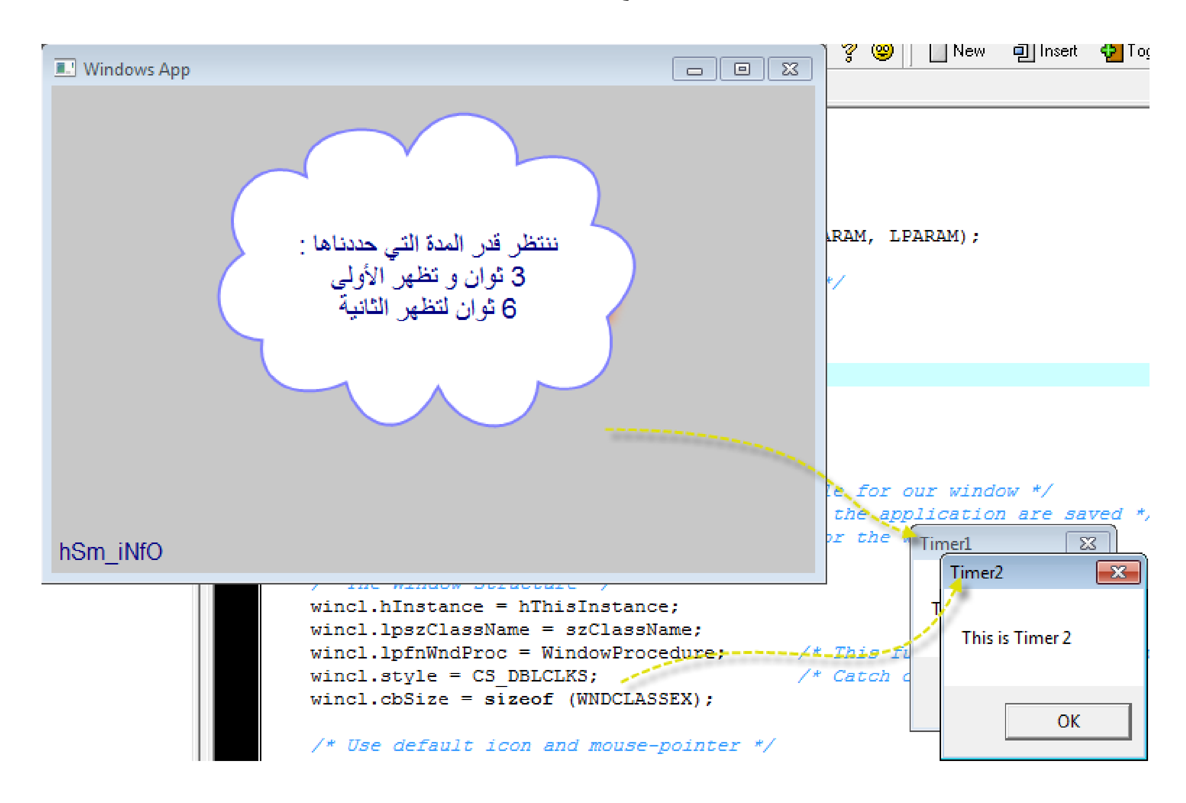
/*  This function is called by the Windows function DispatchMessage()  */
LRESULT CALLBACK WindowProcedure (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)                  /* handle the messages */
    {

        case WM_CREATE:
                SetTimer (hwnd, TIMER_ONE, 3000, NULL) ;
                SetTimer (hwnd, TIMER_TWO, 6000, NULL) ;
                break;

        case WM_TIMER:
             switch (wParam)
             {
                    case TIMER_ONE:
                          MessageBox(NULL,"This is Timer 1","Timer1",MB_OK);
                          break ;
                    case TIMER_TWO:
                          MessageBox(NULL,"This is Timer 2","Timer2",MB_OK);
                          break ;
                           }
                    return 0 ;


        case WM_DESTROY:
            PostQuitMessage (0);       /* send a WM_QUIT to the message queue */
            break;
        default:                      /* for messages that we don't deal with */
            return DefWindowProc (hwnd, message, wParam, lParam);
    }
    return 0;
}
```

الآن مع التجريـب:



منتدى لغات البرمجة - بيت الإبداع

كآخر شيء سأضع بين أيديكم مثالا لإستخدام التايمر في مشروع بسيط من برمجتي:

http://www.startimes2.com/f.aspx?t=20988491

منتدى لغات البرمجة - بيت الإبداع

أرجو أن أكون وفقت في الشرح

¤ هذا الدرس لن تجده في أي موقـع عربي¤

السلام عليكم

منتدى لغات البرمجة - بيت الإبداع

منتدى لغات البرمجة - بيت الإبداع

الكتاب و الشرح من تقديم hSm-iNfO

الإيميل : ab.houssem@hotmail.fr

لأي استفسـار لا تتردد في الإتصال

منتدى لغات البرمجة - بيت الإبداع